UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/807,499	03/24/2004	David John Butcher	550-540	4256

23117 7590 10/18/2010

NIXON & VANDERHYE, PC
901 NORTH GLEBE ROAD, 11TH FLOOR
ARLINGTON, VA 22203

EXAMINER	
LI, AIMEE J	

ART UNIT	PAPER NUMBER
2183	

MAIL DATE	DELIVERY MODE
10/18/2010	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

RECORD OF ORAL HEARING

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE BOARD OF PATENT APPEALS
AND INTERFERENCES

Ex Parte DAVID JOHN BUTCHER, STEPHEN JOHN HILL,
and WILCO DIJKSTRA

Appeal 2009-010086
Application 10/807,499
Technology Center 2100

Oral Hearing Held: August 11, 2010

Before JOSEPH L. DIXON, STEPHEN C. SIU, and JAMES R. HUGHES, *Administrative Patent Judges.*

APPEARANCES:

ON BEHALF OF THE APPELLANT:

STANLEY C. SPOONER, ESQUIRE
Nixon & Vanderhye, P.C.
901 N Glebe Rd., 11th Floor
Arlington, Virginia 22203

The above-entitled matter came on for hearing on Wednesday, August 11, 2010, commencing at 11:22 a.m., at the U.S. Patent and Trademark Office, 600 Dulany Street, Alexandria, Virginia, before Timothy J. Atkinson, Jr., a Notary Public.

THE USHER: Calendar No. 32, Appeal No. 2009-010086, Mr. Spooner.

JUDGE DIXON: Hello, Mr. Spooner. Welcome back.

MR. SPOONER: Yes, I guess.

JUDGE DIXON: Get a double on Nixon and Vanderhye today.

MR. SPOONER: Okay. Long morning.

JUDGE DIXON: Yep. Sorry for keeping you waiting.

MR. SPOONER: What? Not a problem. Not a problem.

JUDGE DIXON: Everybody likes to talk today, I guess.

MR. SPOONER: I guess. Well --

JUDGE DIXON: Or maybe we do.

MR. SPOONER: Well, I have a way to shortcut this, but --

JUDGE DIXON: Excellent. Excellent.

MR. SPOONER: -- I'll try it out on you.

JUDGE DIXON: Okay.

MR. SPOONER: As you know, I'm Stanley Spooner, representing the Assignee of record, Arm Limited. It's a U.K. company. And I want to hit a couple high points from the background of the invention, and this is between pages 2, line 16 to page 3, line 29, and they explain what the difference is between this invention and the prior art. So, it's known to use -- Arm does processors which have -- which support two instruction sets. The arm

instruction set, which are 32-bit instructions, and the thumb instruction set, which are compressed into 16-bit instructions.

It's also known that Arm makes processors that can easily switch between a java state, in which java bite codes are treated as native instructions, and the arm/thumb state. There's a problem switching back and forth, and one known solution to switching is to do what the prior art does. Use a single instruction in which a comparison and branch routine switches program execution from one point to another are combined. In other words, a known compare and branch instruction calculates a target branch using a field within the instruction itself, and that's in the spec at page 2, lines 32 to 34. Obviously, if you're using a field in the instruction, you're limited in the size of that field, and that provides a very limited flexibility, and that's discussed on page 3, lines 2 to 4.

The present inventions solves that problem by providing a compare and branch instruction. Same overall general instruction, but one that uses -- that determines the target branch address from a predetermined -- from a preprogrammed storage value. In other words, that value is stored somewhere else. It's not in the instruction. So, we don't go back to the instruction. We go somewhere else and the decoder in the claims sets out how you find that somewhere else. All right. So, the issue is, does a compare and branch instruction that carries the address within the instruction render obvious a compare and branch instruction that determines the address from a preprogrammed stored valued and a program counter value, which is what the claim does. All right.

The claims briefly -- independent claims 1, 13, 25 and 37; they all recite either a decoder, having certain functionality or a series of function steps in the method claims. Turning right to the Appeal Brief, in section A, the Examiner admits that Ishizaki doesn't teach the copying and determining steps, and we've argued in the brief that she also makes admissions that suggest that it doesn't teach the branching step, or the branching function of the decoder. However, in the Examiner's answer, she then comes in and suggests that the copying step is somewhere disclosed in Ishizaki's abstract. We've gone through it. I think the Reply Brief notes that it's just not there. An important part of the -- and the Reply Brief hasn't been traversed as far as I know by the Examiner in the supplemental or anything like that.

If we go to the Appeal Brief, section B, that's the section which noted that the Hennessy reference doesn't do -- she's alleged that Hennessy teaches the copying step that she previously suggested was missing from Ishizaki, but we noted that Hennessy's copying is in response to an exception occurring, not in response to the comparison result. So, she's not done this comparison and she's not shown how this Hennessy response is in response to that. It's in response to an exception. That's clearly stated in Hennessy, page 411, lines 22 to 25. So, in the obviousness rejection, the Examiner's got the obligation first to point out where the claimed elements, and where the claimed interrelationships are disclosed in amongst the combination of references. In this instance, she's pointed to where structures are shown and certain functions are shown, but she hasn't shown the interrelationship between those functions.

And, if I can digress slightly, I think this case is almost identical to an earlier Board case that I had. Appealing Examiner rejection, and that was Appeal No. 2009-005640, decided January 26th of this year, and that's a case in which Judge Hughes pointed out that the -- well, I'm reviewing --

JUDGE HUGHES: Go ahead.

MR. SPOONER: -- this case and I'm saying, wait a minute I just had this same issue. And so, that's how I think we can shortcut this unless the Board has any serious questions, because in that exact case the Board held that, while the prior art does teach processors, decoders, null value checks and null value exception handlers, as recited in claim 1. The Board properly went further and found that the prior art did not teach a decoder that performs both functions - performing a null value check and controlling branching to a null value exception handler. So that's -- I think that's the situation we have here. There are bits and pieces of the claim that are sprinkled in the two cited prior art references, but there is no interrelationship as set out in the claims.

Now what I think may have contributed to some of the Examiner's confusion here is that -- wait a minute. I missed it. Oh, it's the difference in the meaning of the word exception. Now we mentioned this in the Reply Brief, because that's when I sort of realized it. The Hennessy references states that exceptions are events other than branches or jumps, and that's at page 10, section 5.6, line 4. The Examiner says that Ishizaki teaches that an exception is a conditional branch. That's the Examiner's answer at page 16, lines 4 to 5. Well, if the exception isn't conditional branch, and Hennessy says that an exception is events other than a branch, they don't mesh. So, the

exception in one is not the same thing as an exception in the other. And so, how the Examiner marries the two is simply not apparent.

The second -- so, the first prong of the obviousness is all the claimed elements and interrelationship between elements aren't disclosed in the prior art reference combination. Even if they were the Examiner doesn't meet her burden of providing a clear explicit rationale as to why she's picking and choosing bits from the references, and then combining them in the manner of the claims. We treated that in the Appeal Brief. I don't think we have to belabor it.

Turning to a couple points that were visited in the Reply Brief, error number 2, I think we discussed. The Examiner now is citing some secondary references as teaching the copying step when she didn't previously do that. We couldn't find it, and the Reply Brief goes through that. We couldn't find that in the teaching anywhere. We discussed her failure to provide a motivation in error number 3. We also discussed error number 4, the precedent for the fact that if one of the prior art combinations teaches away from the invention, that combination can't possibly teach the invention. In other words, it rebuts any prima facie case that was made even if there was one made. So in sum, hearing no questions, in summary we've got no prima facie case of obviousness, because the first prong is not met. The claimed interrelationships between the decoder functions are simply not mentioned. The second prong, there's no analysis of her reasons for picking and choosing bits from the two references. So, there is no prima facie case of obviousness, but, even if there was, the whole point of the fact that Ishizaki teaches at column 5, lines 51 to 57, that instructions dot space, dot

space, dot space, dot, are examined to detect the type of exception that has occurred. That teaches the examination of the instruction, which is exactly what the prior art teaches for these compare and branch instructions. You look at the instruction for what's there. That's not what we claim. That would preclude what we're doing, because if you went to the instruction you wouldn't be doing the other stuff that we require. So, no prima facie case for either of the first or second prongs, and, even if there was one it's rebutted by the Ishizaki reference.

JUDGE HUGHES: What was that cite you just --

MR. SPOONER: The --

JUDGE HUGHES: -- for your quote?

MR. SPOONER: -- Ishizaki is at column 5, lines 51 to 57. It says, instructions and then it has some intervening --

JUDGE HUGHES: Right

MR. SPOONER: -- parenthetical inspection -- insertion, but then it says, instructions are examined to detect the type of exception that has occurred. That's the characteristic of the compare and branch instruction that we talked about in the background of the invention that I mentioned when we first went through it.

JUDGE HUGHES: I'm going to make you work for this a little bit, but --

MR. SPOONER: Okay.

JUDGE HUGHES: All right. Starting with claim 1, you've got an -- this is an apparatus claim --

MR. SPOONER: Yes, sir.

JUDGE HUGHES: -- from the preamble, but you've got a lot of looks like method steps in here. So, you've got a lot of functional language.

MR. SPOONER: Right.

JUDGE HUGHES: How does that, the functional language, distinguish this apparatus from prior art?

MR. SPOONER: Well, it's functional language, but it's functional language directed to the decoder. As you may appreciate, there are about eight gazillion ways to put together a decoder, and -- so, the specific circuit in that decoder that does a particular function, that goes a particular place, that stores a particular comparison result, retrieves it -- those circuits aren't significant. It's the whole idea of a decoder, which does -- which is programmed, or created, or hardwired, or a combination of program and hardwire that do these functional interrelationships. And so, the functional language defines what the structure of that decoder has to be. And there are, like I say, many different ways to skin that particular cat. But the end result is you have to have a decoder that does those things. And that 's the interrelationship between those things that the decoder does, is what distinguishes claim 1 from the Ishizaki reference.

JUDGE HUGHES: As far as you're performing a comparison, in Ishizaki -- you're talking about performing a comparison between two registers for doing an exception check. Is that correct -- in general, the portion that's cited by the Examiner.

MR. SPOONER: Well, we're not disputing that Ishizaki teaches a comparison.

JUDGE HUGHES: Right. No, but it's a comparison for an exception check. Is that correct?

MR. SPOONER: I don't think that the claim says exception check.

JUDGE HUGHES: Well, I'm just asking --

MR. SPOONER: Ishizaki does.

JUDGE HUGHES: -- you're characterization of Ishizaki.

MR. SPOONER: Ishizaki does. Yes, sir.

JUDGE HUGHES: All right.

MR. SPOONER: I believe so.

JUDGE HUGHES: So then doesn't Hennessy also teach doing exception checks?

MR. SPOONER: Oh yes. Hennessy's limited to exception checks.

JUDGE HUGHES: All right. And as far as Hennessy goes, you've got multiple registers -- receiving the program counter into an exception program counter. That's disclosed by Hennessy, is it not?

MR. SPOONER: That I don't know, but I'll accept your argument that it is.

JUDGE HUGHES: Okay.

MR. SPOONER: But the important thing to remember is Hennessy doesn't do his copying in response to a comparison result. He does it in response to an exception occurring and the comparison result is --

JUDGE HUGHES: Right, based on an exception check --

MR. SPOONER: -- different from an exception.

JUDGE HUGHES: -- which is the point I'm trying to make. Hennessy does its saving from the program counter to the exception program

counter based on an exception, which is found by doing an exception check, which Ishizaki teaches can be a comparison between two registers. I just want to point out that your language -- the language you're using in the claim doesn't necessarily exclude Hennessy, based on copying from one register to another. You're trying to add something in there that isn't necessarily in the claim.

MR. SPOONER: Right, but again the comparison is done between a value stored in the first register and a value stored in the second register. It's not part of the instruction. The copying is done in dependence upon that comparison of things stored in two registers, and that program counter value is then stored in a third register. It has nothing to do with looking at the instruction that's starts this thing going off. And then we have the determination of the branch address based on the preprogrammed stored value and the program counter value. That's not there.

JUDGE HUGHES: I also wanted to ask you about Hennessy disclosing its -- it discloses a cause register, or a vector address depending on what system you're using for determining what kind of exception is going on?

MR. SPOONER: Where is this?

JUDGE HUGHES: That would be at 412. How exceptions are handled. Let's see -- yeah, 412. You've got a cause register. It's --

MR. SPOONER: Oh, under cause?

JUDGE HUGHES: There's a bullet point there.

MR. SPOONER: Under cause, the bullet point?

JUDGE HUGHES: Yeah. And they also discuss above that -- if you're doing a vector system for handling exceptions -- a vector address, you're -- there's a little table there.

MR. SPOONER: Um-hum.

JUDGE HUGHES: So, aren't those values apart from the instruction itself? The branching instruction that we're talking about through doing an exception handling?

MR. SPOONER: But it's -- from just my cursory review, is there any discussion that the register used to record the cause of the exception stores first and second values, or a program counter value. In other words, are you trying to read that register -- which of the three claimed registers are you trying to read that onto?

JUDGE HUGHES: Right. I'm not trying to read it onto any of them.

MR. SPOONER: Oh, okay. I thought you were, I'm sorry.

JUDGE HUGHES: No. The claim language -- let me pull up -- the Examiner basically cites this information for the determining step. So, you've got a -- requiring that you determining from a branch address -- or, excuse me, determining --

MR. SPOONER: That's end thing.

JUDGE HUGHES: -- a branch address from a preprogrammed stored value and said program countervalue.

MR. SPOONER: Right.

JUDGE HUGHES: So, some value in addition to the program value that's been stored --

MR. SPOONER: Right.

JUDGE HUGHES: -- and the Examiner's pointing to these values as used in that determining step. That's the way I understood the Examiner's argument.

MR. SPOONER: All right. I'm not sure that the vectored interrupts is what we're talking about here, but --

JUDGE HUGHES: No. I'm just pointing out --

MR. SPOONER: Yeah.

JUDGE HUGHES: -- that you've got -- well, that's just one -- the Examiner doesn't clarify exactly what is used for the determining step, but points to page 412, and says that you've got values there.

MR. SPOONER: There are values there.

JUDGE HUGHES: So that's the point I was trying to make. You've got -- let's take, for example, there's a discussion a little farther down on 412 of adding this hexadecimal value for exception handling, and that it is used with a multiplexer to branch to the exception handling address. The area it says, C000000X.

MR. SPOONER: Sub-hex. Yeah.

JUDGE HUGHES: So, that's would be -- that could considered as stored value, could it not, since it's hardwired into a multiplexer?

MR. SPOONER: Oh, sure.

JUDGE HUGHES: Okay. I just wanted to clarify. I don't have any other further questions. Do you guys?

MR. SPOONER: But again, to summarize -- make sure we keep the focus here --

JUDGE HUGHES: I understand.

MR. SPOONER: -- I think that it says there -- the earlier cite to the register, up under the bullet point cause, was recording the cause of the exception. So, that's recording some other sort of data. I'm not sure that that falls within the scope of a program counter value, or that it falls within the scope of a pre-hyphen programmed stored value. And those are the two things that are used to determine the target branch address, in accordance with the claimed decoder. And I just don't see the connection between those so, and maybe you didn't either, but --

JUDGE HUGHES: Well, I'm just trying to clarify where the Examiner was coming from and see if you understood the argument, so I think you've covered it.

MR. SPOONER: I'm not sure I did --

JUDGE HUGHES: All right.

MR. SPOONER: -- but anything else?

JUDGE DIXON: No questions?

MR. SPOONER: Great.

JUDGE DIXON: Thank you for your time.

JUDGE HUGHES: Thank you.

MR. SPOONER: Thank you.

Whereupon, the proceedings, at 11:44 a.m., were concluded.